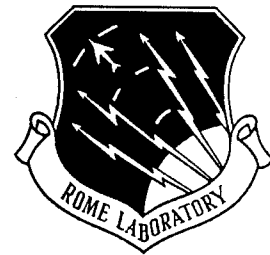


RL-TR-95-277
Final Technical Report
January 1996



TCB SUBSET DBMS ARCHITECTURE PROJECT

Infosystems Technology, Inc.

James P. O'Connor, Mohammed S. Hasan, and Mark S. Smith

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

19960408 127

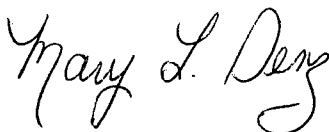
DTIC QUALITY INSPECTED 2

**Rome Laboratory
Air Force Materiel Command
Rome, New York**

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be releasable to the general public, including foreign nations.

RL-TR-95- 277 has been reviewed and is approved for publication.

APPROVED:



MARY L. DENZ
Project Engineer

FOR THE COMMANDER:



JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify Rome Laboratory/ (C3AB), Rome NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE January 1996	3. REPORT TYPE AND DATES COVERED Final ----		
4. TITLE AND SUBTITLE TCB SUBSET DBMS ARCHITECTURE PROJECT		5. FUNDING NUMBERS C - F30602-94-C-0071 PE - N/A PR - R486 TA - 01 WU - P1		
6. AUTHOR(S) James P. O'Connor, Mohammed S. Hasan, and Mark S. Smith		8. PERFORMING ORGANIZATION REPORT NUMBER N/A		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Infosystems Technology, Inc. 6411 Ivy Lane, Suite 306 Greenbelt MD 20770		10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-95-277		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory/C3AB 525 Brooks Rd Rome NY 13441-4505				
11. SUPPLEMENTARY NOTES Rome Laboratory Project Engineer: Mary L. Denz/C3AB/(315) 330-3241				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This report documents the results of an effort to investigate a Multilevel Secure (MLS) Database Management System (DBMS) architecture derived by applying the concepts of Trusted Computing Base (TCB) subsetting as described in the Trusted Database Interpretation of the Trusted Computer System Evaluation Criteria (TCSEC) to a trusted subject MLS DBMS architecture. A TCB subset architecture is a trusted systems architecture in which the overall system security policy is hierarchically partitioned and allocated to different parts (subsets) of the system. Each of these parts implements a reference monitor enforcing the corresponding policy. Each part is similar to a conventional reference monitor, with the exception that it may use the resources of the more primitive subsets (lower in the hierarchy) to enforce its security policy (the most primitive subsets use only the hardware). A subset architecture provides significant benefits in the areas of assurance and evaluability. An alternative to a TCB subset DBMS architecture is a "trusted subject architecture", wherein the DBMS contains some subjects that are not completely constrained by the underlying security kernel. In this report, the design and implementation of a new MLS DBMS architecture that is a hybrid of these two architectures is presented.				
14. SUBJECT TERMS Multilevel secure database management system, Trusted subject, Trusted computing base subset architecture			15. NUMBER OF PAGES 32	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR	

Contents

1	Introduction	1
2	Concepts	2
2.1	TCB Subsets	2
2.2	Trusted Subjects	3
2.3	Relationship Between Subsets and Trusted Subjects	3
3	MLS DBMS Architectures	5
3.1	TCB Subset DBMS Architecture	5
3.2	Trusted Subject DBMS Architecture	6
4	Proposed Architecture	7
4.1	Architecture Definition	7
4.2	Advantages	8
4.3	Disadvantages	9
5	Trusted RUBIX Architecture	10
5.1	Architecture Definition	10
5.1.1	MAC Subset	10
5.1.2	DAC Subset	12
5.2	Satisfaction of Subset Requirements	13
5.2.1	Reference Monitor Requirements	13
5.2.2	Requirements for Evaluation By Parts	14
5.3	Architectural Characteristics	15
5.3.1	TCB Size	15
5.3.2	Performance	15
5.3.3	Resource Utilization	18
6	Design Issues	18
7	Conclusions and Future Research	19

List of Figures

1	Adding a Trusted Subject to a Subset Architecture.	4
2	Abstract Subset Architecture.	7
3	Trusted RUBIX Subset Architecture.	10
4	Trusted RUBIX Process Architecture.	11
5	Trusted RUBIX Domain Isolation Mechanisms.	14
6	Execution Time vs. Selectivity	16
7	Ratio of Execution Time	17

1 Introduction

This document presents the results of a U.S. Air Force Rome Laboratory funded research effort to investigate an MLS DBMS architecture derived by applying the concepts of TCB subsetting as described in the Trusted Database Interpretation [1] of the Trusted Computer System Evaluation Criteria (TCSEC) [2] to a trusted subject multilevel secure DBMS architecture. A TCB subset architecture is a trusted system architecture in which the overall system security policy is hierarchically partitioned and allocated to different parts (subsets) of the system[1]. Each of these parts implements a reference monitor enforcing the corresponding policy. Each part is similar to a conventional reference monitor, with the exception that, it may use the resources of the more primitive subsets (lower in the hierarchy) to enforce its security policy (the most primitive subsets use only the hardware). A subset architecture provides significant benefits in the areas of assurance and evaluability.

In the database security literature, the term "subset architecture" is often used to denote an architecture where the DBMS is completely constrained by an underlying security kernel. Such an architecture was first proposed as part of the SeaView effort [3]. In this architecture, the DBMS runs as one or more untrusted processes on top of a security kernel. This architecture consists of two subsets. The most primitive subset is the underlying security kernel, which is responsible for all mandatory access control (MAC) enforcement. The DBMS forms a second subset which enforces a discretionary access control (DAC) policy on its own objects. While this architecture is an interesting instance of the class of subset architectures, it is not the only possible subset architecture. In fact, the idea of TCB subsets is most powerful when viewed as a general-purpose trusted system design technique.

An alternative to a TCB subset DBMS architecture is a "trusted subject architecture" where the DBMS contains some subjects that are not completely constrained by the underlying security kernel. A trusted subject is an entity (usually a process) that runs with special privilege that allows it to bypass the security policy of an underlying reference monitor. Trusted subjects are employed in a system design when the constraints implemented by the underlying security mechanism make it impossible (or very difficult) to implement required functionality. Since trusted subjects are not completely constrained by the underlying reference monitor, it is crucial that they be carefully analyzed to ensure that they do not violate the intended security policy.

In this report, we present the design and implementation of a new MLS DBMS architecture that is a hybrid of these two architectures. The new architecture was derived by applying the concepts of TCB subsetting to a trusted subject MLS DBMS architecture. This architecture retains many of the strengths of the trusted subject architecture while mitigating its weaknesses. The prototype of this architecture was developed by reengineering the TCB of Trusted RUBIX[4]. Trusted RUBIX is full-functionality SQL2-based MLS client-server relational DBMS that provides mandatory access control at the tuple level. The platform for the prototype is the AT&T 3B2 running UNIX System Laboratories' (USL) UNIX System V Release 4.1 ES (UNIX SVR4.1ES).

Section 2 of this paper presents the concepts of TCB subsets and trusted subjects, and

discusses the relationship between the two. Section 3 discusses MLS DBMS architectures based on each of these concepts and discusses their advantages and disadvantages. Section 4 presents an MLS DBMS architecture that combines the concepts of TCB subsets and trusted subjects, and discusses the advantages and disadvantages of this architecture. Section 5 describes the architecture of the prototype system, discusses how it satisfies the requirements for a subset architecture, and describes its architectural characteristics. Section 6 discusses design issues. Section 7 presents conclusions and future research.

2 Concepts

2.1 TCB Subsets

In a trusted system based on the concept of TCB subsets, the overall system security policy is hierarchically partitioned and allocated to different parts (subsets) of the system. Each of these parts implements a reference monitor enforcing the corresponding policy. Each part is similar to a conventional reference monitor, with the exception that, it may use the resources of the more primitive subsets (lower in the hierarchy) to enforce its security policy (the most primitive subsets use only the hardware). A subset architecture can be incrementally evaluated, in that each of the parts can be separately evaluated against their respective policies. The evaluation of a given part depends upon the evaluation of the more primitive subsets which it uses. Even though the parts can be incrementally evaluated, it still must then be argued that when composed, parts enforce the original system security policy.

The idea of having multiple levels of security kernels, each implementing a security policy on its own objects, dates back to the design of the UCLA Virtual Machine System [5]. The current concept of TCB subsets grew out of work on the concept of extensible TCBs [6] and the first full treatment of this form of the concept was published in [7]. Here, the basic idea was to generalize the reference monitor concept [8] to support the goal of incremental evaluation of trusted systems. The Trusted Database Interpretation (TDI) [1] of the Trusted Computer System Evaluation Criteria (TCSEC) [2] embraced the concept of *hierarchically* related subsets as a basis for trusted DBMS development and evaluation.

The TDI formally defines a subset M as the a set of software, firmware, and hardware (where any of these three could be absent) that mediates the access of a set S of subjects to a set O of objects on the basis of a stated access control policy P and satisfies the properties:

1. M mediates every access to objects in O by subjects in S;
2. M is tamper resistant; and
3. M is small enough to be subject to analysis and tests, the completeness of which can be assured.

Furthermore, the TDI specifies a set of conditions that a subset architecture must meet in order to be eligible for an evaluation by parts. These conditions are:

1. The candidate TCB subsets are identified;
2. The system policy is allocated to the candidate TCB subsets;
3. Each candidate TCB subset $M[i]$ includes all the trusted subjects with respect to its technical policies $P[i]$;
4. The TCB subset structure is explicitly described;
5. Each TCB subset occupies distinct subset-domains; and
6. The more primitive TCB subsets provide support for the reference validation mechanism arguments for the less primitive TCB subsets.

Any architecture that claims to be a subset architecture must satisfy these conditions.

2.2 Trusted Subjects

A trusted subject is an entity (usually a process) that runs with special privilege that allows it to bypass the security policy of an underlying reference monitor¹. For example, UNIX System V Release 4.2 ES allows a subject to be granted a number of privileges, including: read-up (MACREAD), write-down (MACWRITE), and modify process level (SETPLEVEL) [9]. Other operating systems provide a more granular privilege mechanism where processes are only trusted within a range [10].

Trusted subjects are employed in a system design when the constraints implemented by the underlying security mechanism make it impossible (or very difficult) to implement required functionality. They may be used as part of the implementation of a multilevel secure operating system (and hence are evaluated as part of the operating system evaluation), or they may be added later to support a trusted application (e.g., a guard application). Since trusted subjects are not completely constrained by the underlying reference monitor, it is crucial that they be carefully analyzed to ensure that they do not violate the intended security policy.

2.3 Relationship Between Subsets and Trusted Subjects

The first issue that must be addressed is whether it is possible to apply the concept of TCB subsets to an architecture utilizing trusted subjects. One way to illustrate that the concepts of trusted subjects and TCB subsets are compatible, is to start with a valid TCB subset architecture, add a trusted subject, and argue that the result can be made into a valid (but different) subset architecture. A subset architecture is valid if it satisfies the six criteria for a subset architecture, and each subset possesses the three reference monitor properties required of a subset.

¹Even though most discussions of trusted subjects focus on the ability to circumvent mandatory access control, a process can be trusted with respect to any aspect of the policy implemented by the reference monitor (e.g., discretionary access control).

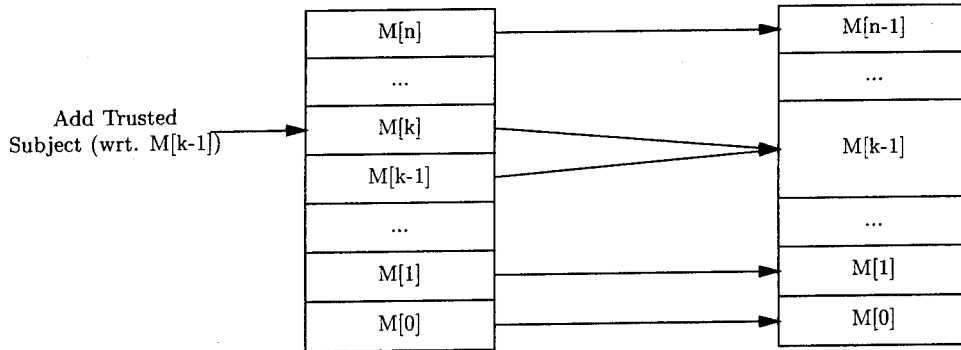


Figure 1: Adding a Trusted Subject to a Subset Architecture.

Assume that there exists a TCB that is layered into n hierarchical subsets $M[0]$, $M[1]$, ..., $M[k]$, ..., $M[n]$, and is valid by the above definition. Now suppose that we add a trusted subject to subset $M[k]$ (trusted with respect to subset $M[k-1]$). This situation is shown on the left side of Figure 1. As modified, this architecture is not valid because it violates condition 3. However, we can combine the subsets $M[k]$ and $M[k-1]$ into a single subset that is allocated the combined policies of the two subsets.² The resulting architecture, which is shown on the right side of Figure 1, is a new candidate subset architecture.

It should be apparent that the candidate architecture satisfies the six conditions for a subset architecture as listed above. The remainder of this section argues that the new subset ($M[k-1]$) can satisfy the required reference monitor properties as well. The first property can always be satisfied because the new subset can use the same mechanisms as the old $M[k]$ and $M[k-1]$ subsets to ensure that it is not bypassed. The second property can always be satisfied because the new subset can use the same mechanisms as the old $M[k]$ and $M[k-1]$ subsets to ensure that it is tamper resistant. The satisfaction of the third property depends on characteristics of the original subsets that were combined. As noted in the TCSEC, the third property is currently interpreted to mean that the TCB “must be of sufficiently simple organization and complexity to be subjected to analysis and tests, the completeness of which can be assured” [2]. It certainly can be argued that if the original two subsets satisfied this criterion, and the additional trusted subject satisfied this criterion, then the new subset would satisfy it. At higher assurance levels there is also the implication that modules that are not protection critical have been excluded from the TCB. A similar argument could be made that if the original two subsets were in some sense “minimal” given their respective policies, the combined subset would also be minimal for the combined policy, provided that the trusted subject itself were minimal.

The above argument demonstrates that, although trusted subjects have a definite impact on subset architectures, they can not be determined a priori to be incompatible concepts. The implication is that an MLS DBMS can be implemented using trusted subjects and may still derive benefit from an application of the concept of TCB subsets.

²In general, a trusted subject introduced in subset $M[i]$, that is trusted with respect to level $M[j]$, $i > j$, will require all subsets $M[k]$, $i \geq k \geq j$, to be combined.

3 MLS DBMS Architectures

3.1 TCB Subset DBMS Architecture

The concept of TCB subsets has been applied in the domain of database architectures to produce a TCB subset DBMS architecture. This architecture was first proposed as part of the SeaView effort [3]. In this architecture, the DBMS runs as one or more untrusted processes on top of a security kernel. This architecture consists of two subsets. The most primitive subset is the underlying security kernel, which is responsible for all mandatory access control enforcement. The DBMS forms a second subset which enforces a discretionary access control (DAC) policy on its own objects.

The advantages of this architecture are ease of evaluation and assurance. The ease of evaluation of this architecture is due to the fact that, since this architecture has no trusted subjects, the DBMS is prevented from doing anything that would invalidate a previous evaluation of the underlying security kernel. There is no need to perform any re-evaluation of the underlying operating system. The mandatory assurance characteristics of this architecture are derived directly from the mandatory assurance characteristics of the underlying operating system. For this reason, the mandatory assurance level of the DBMS should be the same as that for the underlying security kernel.

The disadvantages of this architecture are that it is inflexible, difficult to use, difficult to implement, and inefficient. The architecture is inflexible because:

1. *Polyinstantiation is unavoidable.* This is because the presence of similarly named objects at higher or non-comparable levels cannot be detected. This is true not only of tuples, but of databases, relations, and schemata.
2. *Integrity constraints cannot always be enforced.* This is because the enforcement of certain constraints require the ability to detect the presence of objects at a higher or non-comparable level or to remove objects at a lower or non-comparable level.³
3. *DBMS Trusted subjects cannot be supported.* This is because the DBMS itself cannot circumvent MAC privileges, therefore it cannot offer any such services to its clients.
4. *Information cannot be downgraded.* This is because downgrading requires a trusted subject, which is not permitted in a TCB subset DBMS architecture.

The resulting DBMS is difficult to use for the above reasons as well as the fact that database dumps, restores, and bulk loads must be performed at each level in the security lattice. The implementation is difficult because:

³It can be argued that this and the previous "disadvantage" are actually necessary characteristics of a secure system (because both failure to support polyinstantiation and complete enforcement of integrity constraints can introduce covert channels). The issue is that a system based on this architecture cannot give the DBA the option to trade-off security and data integrity.

1. *Data must be fragmented.* This is because the DBMS must store all multilevel data, metadata, and log information in the single level objects provided by the security kernel.
2. *Concurrency control and recovery must be performed without global knowledge.* Since the subjects that perform these operations are necessarily single level, they can only see the portion of the relevant data that they dominate.
3. *DBMS processes must be replicated at each security level.* Since the DBMS subjects are necessarily single level, there must be one for each client security level to be supported.

Finally, the architecture is less efficient because:

1. *The amount of I/O is increased because of data/log fragmentation.* This is because the DBMS subject must read from one file for each level in the security level lattice that it dominates. This will significantly reduce the effectiveness of buffering.
2. *The DBMS must rely on the operating system file management.* Database management systems frequently implement their own file systems that are optimized for database access. Since these file systems are necessarily multilevel, they cannot be implemented using untrusted subjects.
3. *Each DBMS process must be duplicated at each security level.* As noted above, the DBMS TCB subset architecture requires the duplication of processes by security level. This will have a negative impact on performance because of the additional context switching overhead (and resource consumption).

3.2 Trusted Subject DBMS Architecture

In the trusted subject DBMS architecture, the DBMS includes one or more subjects that are trusted with respect to the security policy of the underlying operating system. The composed system (operating system plus application) implements a security policy that is potentially different from the one originally implemented by the operating system.

The advantages of this architecture are performance, flexibility, ease of implementation, and ease of use. The primary disadvantages of this architecture are low assurance and evaluation difficulty. Both of these disadvantages are a result of the fact that, since the DBMS is not fully constrained by the underlying operating system TCB, flaws in its implementation can cause a breach of mandatory security. The difficulty of evaluating such a system is compounded by that fact that the combination of the trusted subject and the TCB of the underlying operating system can introduce information flows that cannot be discovered by performing an analysis of the trusted subject alone. The implication of this is that it is not sufficient to look at the trusted subject alone when evaluating the security characteristics of the DBMS. At least some of the evaluation of the underlying operating system must be repeated.

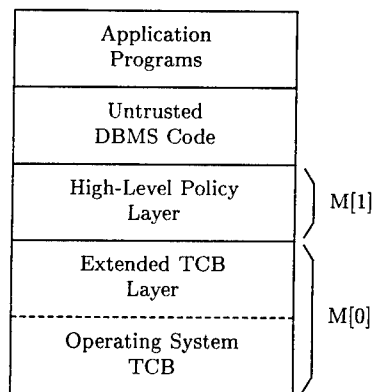


Figure 2: Abstract Subset Architecture.

4 Proposed Architecture

The trusted subject and TCB subset architectures presented in the previous section are generally considered to be disjoint, each having its own distinct advantages and disadvantages [11]. This view is not consistent with the idea of TCB subsets as a general-purpose design technique. As discussed in Section 2.3, there is no technical reason why the concept of TCB subsets cannot be productively applied within the domain of trusted subject DBMS architectures. This section presents an MLS DBMS architecture derived by applying the concept of TCB subsets to a trusted subject DBMS architecture.

4.1 Architecture Definition

Figure 2 shows an abstract MLS DBMS architecture that was derived by applying the concept of TCB subsets to a trusted subject DBMS architecture. This architecture consists of two subsets, $M[0]$ and $M[1]$. The $M[0]$ subset enforces a mandatory access control policy on DBMS objects (e.g., tuples) and consists of the operating system TCB combined with the minimal amount of trusted DBMS code required to implement the desired policy. The $M[1]$ TCB subset is layered upon $M[0]$ and enforces a discretionary access control policy that is a refinement of the policy enforced by the $M[0]$ TCB. Each of these subsets must be isolated via a domain isolation mechanism (e.g., protection rings [12]).

This architecture is abstract in the sense that it can describe a wide range of actual MLS DBMS systems. One important detail that has been omitted is the actual allocation of DBMS functionality to the subsets. For the $M[0]$ subset, some possibilities include: no DBMS functionality (the conventional TCB subset case), a simple filter, and significant DBMS functionality (e.g., access methods and scheduler). For the $M[1]$ subset, some possibilities include: a null subset (i.e., rely on the DAC, if any, provided at the $M[0]$ subset) and subsets whose functionality is determined by the granularity of the definition of protected objects in the system's security policy (e.g., relations, columns, views).

The $M[0]$ subset of this architecture actually consists of two "parts" (in the TDI sense)

that are isolated in separate protection domains. Since one of these parts contains subjects trusted with respect to the other, these two parts do not qualify for an evaluation by parts. As noted in the TDI, even though these parts do not qualify for an evaluation by parts, it is likely that significant savings can be recognized by reusing the results of the evaluation of the underlying security kernel. The problem encountered here is that there is no theory that can be used to quantify these savings a priori.

It is also worth noting that the $M[1]$ subset (or the $M[0]$ subset for that matter) could be further subdivided into additional subsets if desired. This would of course depend on a meaningful decomposition of the security policy and the availability of the required domain isolation mechanisms.

The remainder of the paper will focus on an instance of the above architecture in which the extended TCB layer includes the *minimal* DBMS functionality required to retain the significant advantages of the trusted subject architecture as discussed in 3.2.

4.2 Advantages

The proposed architecture retains the significant advantages of the trusted subject DBMS architecture while mitigating its disadvantages. The advantages are retained through the judicious use of trusted subjects (e.g., to avoid data fragmentation). The disadvantages are mitigated by isolating all DBMS code that requires mandatory privilege to the lowest level subset. The result is a system that offers significantly higher assurance for mandatory access control and is more evaluable than a similar system with a monolithic TCB. The assurance advantages are a result of the fact that:

- *The amount of code that can cause a violation of MAC is significantly decreased.* The amount of DBMS code in the $M[0]$ subset is significantly less than that in the TCB as a whole *and* only subjects in this subset can run with special MAC privileges. Since subsets must satisfy the isolation and non-bypassability requirements for a reference validation mechanism, these properties guarantee that only code in the $M[0]$ subset can cause a violation of MAC.
- *The effectiveness of assurance techniques is increased.* Assurance techniques are more effectively applied at a lower level of abstraction. Since assurance techniques must be applied to each subset, the TCB subsets approach forces you to apply these techniques more directly to the portions of the system responsible for MAC enforcement (viz., the $M[0]$ subset). Additionally, if you subscribe to the notion of balanced assurance [13], this approach has the effect of focusing your assurance efforts where they will have the most impact.

The proposed architecture is easier to evaluate because:

- *The scope of global analysis is reduced.* Developing a system using trusted subjects requires that certain global analysis be performed on the combined underlying TCB and the DBMS TCB (e.g., covert channel analysis). If the DBMS TCB has a multi-subset TCB, only the $M[0]$ TCB must be considered in these global analyses.

- *The evaluation task can be partitioned.* One of the primary benefits of the TCB subsets approach is the ability to divide a complex system into parts and evaluate the parts incrementally. This approach makes the evaluation of a complex TCB more tractable.
- *The re-assessment of modified or ported systems is simplified.* A TCB subset architecture has the characteristic that the evaluation impact of certain changes is isolated to the subset in which they occur. This can result in significant savings in the area of re-assessment.
- *Subsets can be evaluated to different assurance levels.* This architecture has the characteristic that the different subsets can be evaluated to different assurance levels. That is, the M[0] subset could be evaluated to a relatively high level (e.g., B3 or A1) while the M[1] subset could be evaluated at a lower level (e.g., C2).⁴

In addition to the assurance and evaluation benefits, applying the concept of TCB subsets to a trusted subject architecture permits a vendor to support a family of MLS DBMS products without duplicating evaluation effort. A vendor could support an entire product line (e.g., with products supporting different DAC policies) with the basic M[0] TCB at its core. Since a subset architecture allows incremental evaluation, the underlying M[0] TCB need only be evaluated once for the entire product line.

4.3 Disadvantages

The application of the concept of TCB subsets to trusted subject DBMS architectures has some disadvantages as well. Specifically, the proposed architecture has the following disadvantages:

- *Implementation difficulty associated with multiple protection domains.* The architecture presented above will require at least four hierarchical protection domains: one for the operating system, one for the extended TCB layer, one for the M[1] subset, and one to protect the integrity of the untrusted DBMS code. These domains can be provided through a variety of mechanisms and each domain need not use the same mechanism.
- *Performance overhead associated with multiple protection domains.* As noted above, this architecture requires at least four hierarchical protection domains. Crossing domain boundaries is likely to have a negative impact on DBMS performance.
- *Reduced Flexibility.* This reduction in flexibility occurs because the M[1] subset cannot violate policy of the M[0] subset even in cases where it would be desirable. For example, it would not be possible to support trusted stored SQL procedures in a

⁴Current evaluation practice is to require all subsets to be evaluated at a uniform assurance level. There is, however, no technical reason to require a uniform assurance level provided that a given subset does not depend upon a less assured subset.

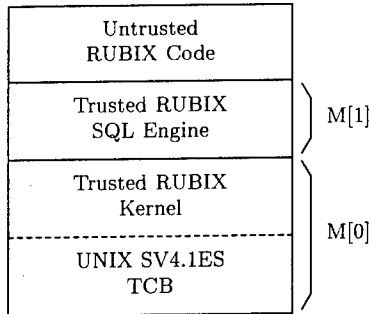


Figure 3: Trusted RUBIX Subset Architecture.

DBMS in which SQL is outside of the mandatory subset. Exactly how much flexibility is lost is determined by what DBMS functionality is placed in the M[0] subset.

These disadvantages are significantly less than those realized in the conventional TCB subset DBMS architecture.

5 Trusted RUBIX Architecture

5.1 Architecture Definition

The Trusted RUBIX TCB architecture is shown in Figure 3 and consists of two subsets: a MAC subset (M[0]), which is responsible for implementing a mandatory security policy, and a DAC subset (M[1]) which is responsible for implementing the discretionary security policy.

5.1.1 MAC Subset

The MAC subset consists of the operating system kernel and the Trusted RUBIX kernel which runs as a trusted subject on the underlying operating system. The policy implemented by the MAC subset is a Bell and LaPadula [14] with databases, relations, and tuples as objects. The DBMS functions implemented in the MAC subset are:

- File Management.
- Buffer Management.
- B-Tree Management.
- Relation Management.
- Transaction Management.
- Audit of MAC operations.

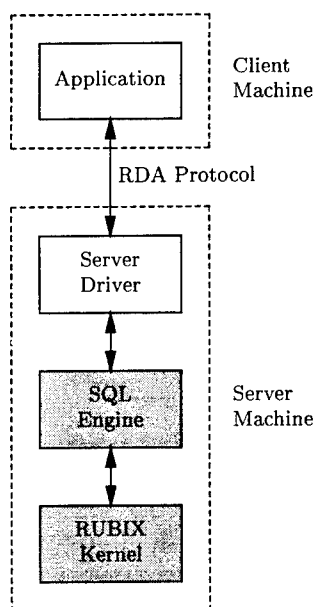


Figure 4: Trusted RUBIX Process Architecture.

The architecture of the MAC subset⁵ is based on the concept of a *protected subsystem* [15]. All MAC protected data are stored in one or more volumes, which are single-level operating system objects. To support fine-grained multilevel objects (viz., tuples), labels are attached to individual database items within each operating system object. Note that these labels are DBMS labels and not operating system labels—the operating system views these labels strictly as data and attaches no security significance to them. The DBMS is trusted to properly associate and maintain the label of each item and to correctly interpret those labels so that, in cooperation with the operating system kernel, the security policy can be correctly enforced.

The Trusted RUBIX portion of the MAC subset is implemented as a separate operating system process as shown in Figure 4. The resources protected by the MAC subset are protected from external access by labeling them with the reserved security level `USER.RUBIX` (an alias for `user:rubix`). Processes in the MAC subset set their process level to this security level on invocation which allows them to access the protected resources. To do this, these processes must be given the `UNIX SETPLEVEL` (set process level) privilege. This privilege is explicitly acquired immediately before the `lvlproc()` system call, and released immediately after the call completes.⁶ There are two notable characteristics of the `USER.RUBIX` level.

⁵For convenience, we will use the term “MAC subset” to refer to the Trusted RUBIX kernel portion of the `M[0]` subset, although technically the `M[0]` subset consists of the Trusted RUBIX kernel, the underlying operating system kernel, and the hardware.

⁶This technique is known as *privilege bracketing* and is employed throughout the implementation. The motivation for privilege bracketing is to minimize the execution time during which a trusted process holds a privilege, thereby supporting the *least privilege* principle within the DBMS TCB.

First, it includes the category *rubix*, which is reserved for subjects and objects in the MAC subset. Since processes outside of the MAC subset cannot have the rubix category in their label, they are prevented from directly accessing the MAC subset subjects and objects. Second, it does not contain the category *login*, which is part of the labels of all untrusted UNIX subjects and objects. Since MAC subset subjects do not have the login category in their level, they are prevented from accessing UNIX subjects and objects. Thus, the design of Trusted RUBIX uses the underlying UNIX mandatory access control mechanisms to isolate M[0] subjects and objects from UNIX subjects and objects.

The executables that make up the MAC subset are MAC protected from unauthorized modification by labeling them with the hierarchical level USER_PUBLIC, which is dominated by the level of all untrusted processes. The *-property enforced by the operating system thereby protects these programs from unauthorized modification. In addition, all MAC subset programs are installed with appropriate UNIX DAC permissions that prevent unauthorized access.

Since the MAC subset is implemented as a separate process its interface is an inter-process communication (IPC) interface. Conceptually, this interface consists of a set of procedures. From a client's point of view, all that is necessary to access the services of the MAC subset is to link to the library containing these procedures. These procedures are implemented as a form of remote procedure call. There are two versions of each procedure: a client-side procedure that runs in the DAC subset domain and a server-side procedure that runs in the MAC subset domain. Each client-side call to an interface function causes a peer function to be executed in the MAC subset domain. When an initialization routine is called, the server (rxkernel) process is invoked, and a UNIX pipe is established between the client and server process for synchronous communication. When a program calls one of the client-side access functions, the arguments to the call are collected and passed, along with a procedure identifier to the server side. The client procedure then blocks, and waits for a response. On the server side, a dispatcher function examines the procedure identifier and calls the appropriate server-side procedure with the communicated arguments. The server-side procedure validates its arguments, and then performs its function. When this call returns, the dispatcher function sends back any return values to the client procedure which then returns like a normal procedure call. The dispatcher function then blocks, awaiting the next request. The communication channel is brought down by a termination routine.

5.1.2 DAC Subset

The DAC subset consists of the Trusted RUBIX SQL engine. This subset depends upon on the MAC subset and implements a discretionary policy that is a further restriction of the policy enforced by the MAC subset. The DAC subset implements a DAC policy on databases, schemata, relations, views, indexes, and columns. The DBMS functions implemented in the DAC subset are:

- Query Parsing.
- Query Optimization.

- Execution of Query Plans.
- Join Algorithms, Sort Algorithms, Group-By, etc.
- Integrity Constraints.
- View Management.
- Index Management.
- Audit of DAC objects.

The architecture of the DAC subset is also based on the concept of a protected subsystem. Similar to the MAC subset, the DAC subset is implemented as a separate process. The DAC subset protects its resources using the underlying trusted UNIX DAC mechanism. All DAC subset resources (including database volumes) are protected from external access by making them accessible only to subjects in the reserved UNIX group rubixTP. Once these protections are in place, the underlying operating system will not permit access to the resources unless the effective group-id of the accessing process is rubixTP. Untrusted processes cannot have this group-id because rubixTP is a reserved group with no members. The UNIX setgid (set group identifier) mechanism is used to set the effective group-id of the DAC subset process to rubixTP upon invocation. This permits the process access to the protected resources. Figure 5 summarizes the mechanisms that are used to insure the non-bypassability of the two subsets.

The DAC subset uses the same mechanism as the MAC subset to protect its executables from tampering. The interface to the DAC subset is an IPC interface similar to that used in the MAC subset.

5.2 Satisfaction of Subset Requirements

5.2.1 Reference Monitor Requirements

Each TCB subset must satisfy the three reference monitor requirements that were listed in Section 2.1. The first requirement states that the reference monitor cannot be bypassed, that is, that subjects external to the subset cannot access protected resources without having that access mediated by the subset. This requirement is satisfied using different mechanisms for the MAC and DAC subset. The MAC subset prevents external subjects from accessing its data by labeling them at the level USER_RUBIX. The DAC subset prevents external subjects from accessing its data by storing them such that only members of the reserved group rubixTP can access them. Since only processes in the DAC subset are permitted membership in this group, this prevents external subjects from accessing these data.

The second requirement for a reference monitor is that it is tamper resistant. Both the MAC and DAC subset use the same mechanisms to protect themselves. Since MAC and DAC subjects run as separate processes, they are protected from tampering via the underlying operating system's process isolation mechanism. The MAC subset processes

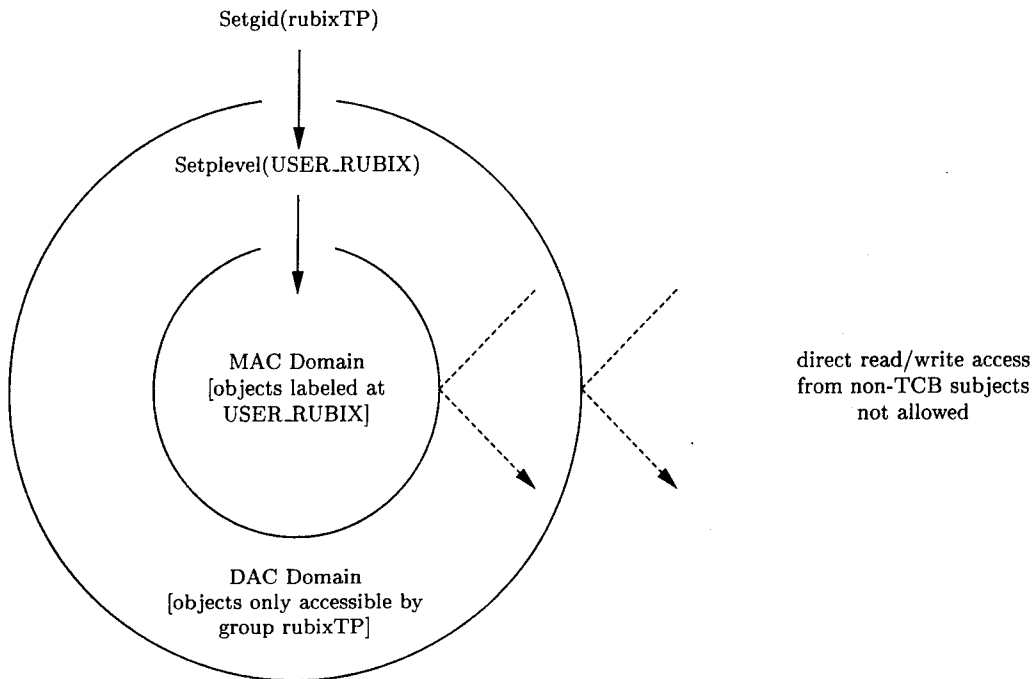


Figure 5: Trusted RUBIX Domain Isolation Mechanisms.

have an extra measure of protection because they run at the level `USER_RUBIX` and are therefore further isolated from untrusted processes.

The third requirement for a reference monitor is that it is small enough to be subject to analysis and tests, the completeness of which can be assured. To satisfy this requirement, the Trusted RUBIX MAC and DAC subsets have been designed to be modular and small enough so that correctness may be established. The TCB subset approach allows a divide and conquer approach to be used in establishing the correctness of the composite Trusted RUBIX TCB.

5.2.2 Requirements for Evaluation By Parts

This section describes how the Trusted RUBIX TCB architecture satisfies the conditions for an evaluation by parts as specified in 2.1.

- *The candidate TCB subsets are identified.* The two subsets that comprise the Trusted RUBIX TCB are shown in Figure 3. The top level specifications for these subsets are contained in [16].
- *The system policy is allocated to the candidate TCB subsets.* The Trusted RUBIX security policy is described in [17]. This policy is layered into a MAC and DAC portions, which are allocated to the MAC (`M[0]`) and DAC (`M[1]`) subsets, respectively.

- *Each candidate TCB subset $M[i]$ includes all the trusted subjects with respect to its technical policies $P[i]$.* The only trusted subjects in the architecture are those in the Trusted RUBIX Kernel that are trusted with respect to the underlying operating system. Since the Trusted RUBIX Kernel and operating system are part of the same subset, this condition is met.
- *The TCB subset structure is explicitly described.* The subset structure is shown in Figure 3. It is a strictly hierarchical structure with $M[1]$ depending on $M[0]$.
- *Each TCB subset occupies distinct subset-domains.* The domain isolation mechanism used by the two subsets was described in Sections 5.1.1 and 5.1.2 and illustrated in Figure 5.
- *The more primitive TCB subsets provide support for the reference validation mechanism (RVM) arguments for the less primitive TCB subsets.* Section 5.1.2 discusses how the $M[1]$ subset uses the services of $M[0]$ to support its reference validation mechanism. Furthermore, even though the Trusted RUBIX Kernel is not a separate subset, it uses the services of the UNIX SVR4.1ES operating system to support its reference validation mechanism.

5.3 Architectural Characteristics

5.3.1 TCB Size

As shown in Figure 3, Trusted RUBIX can be viewed as consisting of three major components: the Trusted RUBIX Kernel, the SQL Engine, and Untrusted Code. The Trusted RUBIX Kernel consists of 20K lines of code, the SQL Engine consists of 58K lines of C code, and there are 16K lines of untrusted code. All of these numbers are source lines of C code excluding comments. It is worth noting that the number of lines of code that run with MAC privilege has been reduced by 73 percent.

5.3.2 Performance

It was anticipated that the subset TCB architecture would exhibit degraded performance when compared with the monolithic TCB architecture. The primary reason degraded performance was anticipated is that each fetch of a tuple requires a context switch between the MAC and DAC subsets. A secondary reason degraded performance was anticipated was that data must be serialized and copied across the interface for each tuple accessed.

To get a measure of relative performance, we ran a benchmark on two different versions of Trusted RUBIX. These two versions differ in how they were linked. The first version was linked with a monolithic TCB (i.e., both subsets in the same address space). In this version the interface between the DAC and MAC subsets was a procedure call interface. The second version was linked with a subset TCB (i.e., the MAC and DAC subsets in different address spaces). In this version, the interface between the DAC and MAC subsets was a UNIX pipe.

The benchmark runs against a subset of the database specified in the Wisconsin Benchmark. The database consists of one table (THOUSKTUO1) containing 100K tuples. This table contains two integer columns: unique1D and unique2D. The unique1D column is the primary key, and contains values that monotonically increase from 1. The non-key column unique2D also contains values that range from 1 to the cardinality of the table, but these values are randomly distributed.

The query used is of the form:

```
SELECT unique1D
FROM THOUSKTUP1
WHERE uniqueID < CONSTANT
```

This query selects different numbers of tuples from the beginning of the relation depending on the value of CONSTANT. For example, the selection predicate "*unique1D* < 1000" selects the first 1000 tuples of the relation, in a contiguous sequence. Varying the constant allows the number of tuples selected to be controlled. This allows us to compare the impact of selectivity on the two architectures.

Figure 6 shows the impact of selectivity on execution time. This diagram shows that as the selectivity of the query increases, so does the difference in execution time. This is due to the fact that the performance penalty in the subset architecture is due to the cost of transferring tuples across the subset boundary, and this only occurs if the tuple is selected.

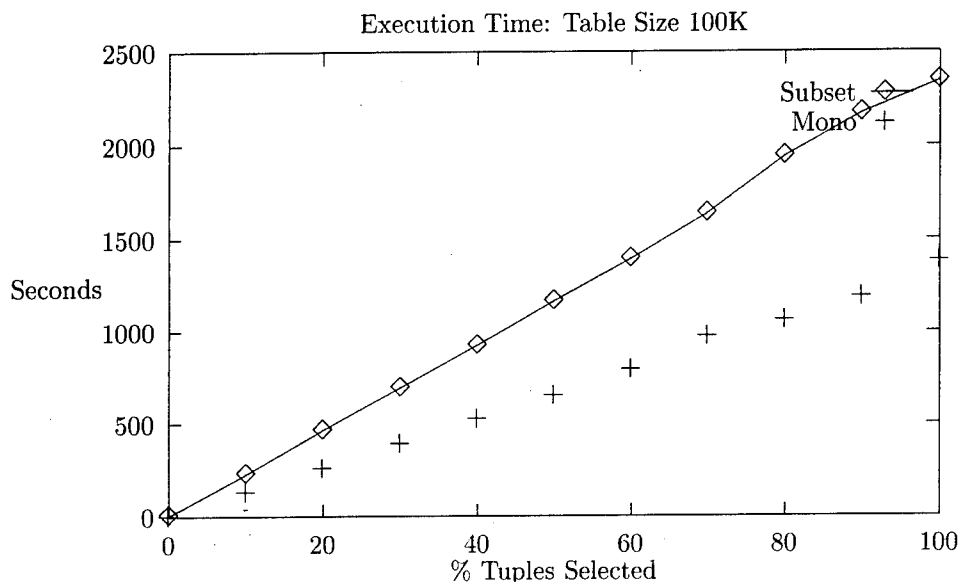


Figure 6: Execution Time vs. Selectivity

Figure 7 shows the ratio of the execution time for monolithic and subset TCBs against

the selectivity (% tuples selected). The formula for the ratio is

$$ratio = \left(\frac{\text{time for subset TCB(seconds)}}{\text{time for monolithic TCB(seconds)}} - 1 \right) \times 100$$

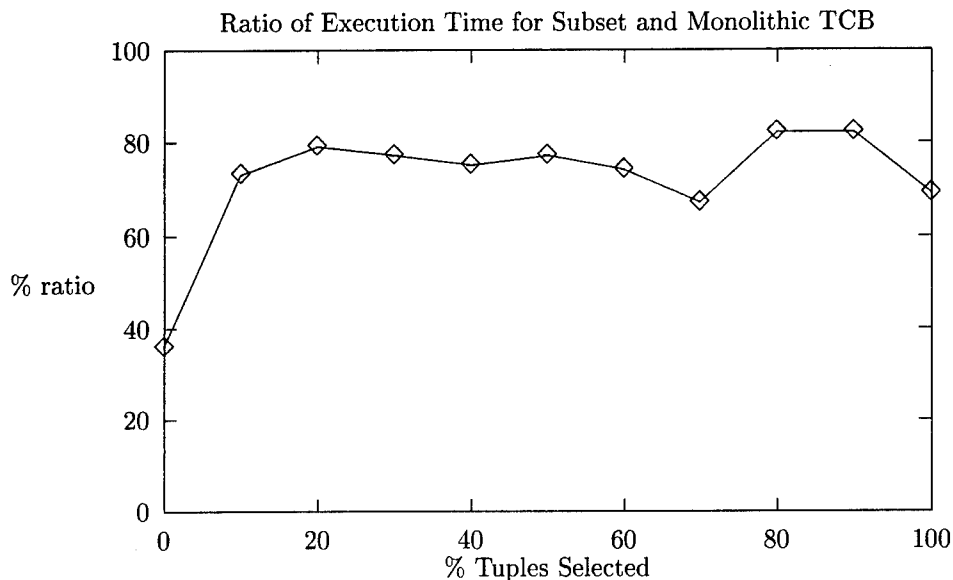


Figure 7: Ratio of Execution Time

This ratio gives the difference in execution time as a percentage of the execution time for the monolithic TCB. For example, when the selectivity is 40% for table size 100K, subset TCB takes more time, and the difference is 75% of the time taken by the monolithic TCB. This ratio starts out at 35% when a relatively small number of tuples is selected. This will be the cost of the subset architecture for most queries. As the number of tuples returned increases, this ratio rises. As the number of tuples reaches 10-15K the ratio stabilizes at around 60-75%. This is the point at which the cost of retrieving tuples dominates all other costs.

There are a number of performance enhancements that we could incorporate into the design that we believe would significantly improve performance. These are:

- *Multi-record return.* Currently the domain boundary between the DAC and MAC subset is crossed for every tuple returned. Each of these domain crossings necessarily causes a context switch at the operating system level. If we modify the interface so that tuples are prefetched so that multiple tuples are transferred on a single context switch, the number of context switches can be reduced drastically. This is significant because we believe that most of the time being lost in the new architecture is a result of context switches.

- *Streamlined record structure.* Another possible optimization is to streamline the structure used to pass tuples across the MAC/DAC interface. The current structure includes both the tuple data and metadata. In many cases the meta data is significantly larger than the actual tuple data. All of this information must be copied across the interface for each tuple accessed. Separating the metadata from the tuple data could result in a significant reduction in the amount of time spent copying data.
- *More efficient IPC mechanism.* The current implementation uses a UNIX pipe for the interface between the MAC and DAC TCBs. Changing to a more efficient IPC mechanism (e.g., shared memory) could also improve performance. Before this change can be made a number of issues related to the granularity of protection on UNIX IPC objects must be resolved.

5.3.3 Resource Utilization

The subset architecture version of Trusted RUBIX requires additional resources over those required in the monolithic TCB version. This increased resource utilization is the result of the fact that the subset version uses the UNIX SV4.2ES process isolation to isolate the two subsets. The difference is that in the subset version, there are two UNIX processes for each instance of the server rather than one. The additional resources required are the memory needed to store the additional process context, the processor usage required for the context switch between the two processes, the processor usage required to move data between the two processes, the disk space needed to swap out the processes, and the disk space needed to store the separate executables.

6 Design Issues

The major design issues that were faced in re-engineering the TCB architecture of Trusted RUBIX were:

- *The location of the division between the two subsets.* There were four primary factors that affected this decision. The first factor was the design of the monolithic TCB. The second factor was the partitioning of the security policy. The combination of these first two factors suggested a natural point at which to partition the system. The third factor was a desire to minimize the size of the MAC subset. The fourth factor was the goal to minimize the impact on the performance and flexibility of the architecture. The final location of the division was arrived at by starting at the position suggested by the first two factors, and then moving it down until the tradeoff of the third and fourth factors caused us to stop.
- *The mechanism to be used to support subset hierarchical domains.* The domain isolation approach used in the architecture is described in Section 5.1.1. The major

factor in choosing the approach we did was the support provided by the underlying secure operating system. On a different platform, a different domain isolation mechanism would probably be used (e.g., protection rings [12]).

- *The mechanism to be used for cross-domain communication.* Given the domain isolation mechanism that was selected, there were a number of approaches that could be used for cross-domain communication (e.g., UNIX pipes, shared memory, message queues). UNIX pipes were selected because they were the only IPC mechanism that could be exclusively shared between two processes—the others were protected at the granularity of the user/group. The rationale for this was that it would be easier to argue the correctness of an IPC interface that was guaranteed to have a fixed pair of communicants.
- *The storage of temporary results.* In answering certain types of queries, Trusted RUBIX needs to store temporary results in relations. The problem that occurs is that normal MAC rules require labels on these temporary results to float up to the level of the subject. This will result in overclassification of the result. For example, suppose a secret subject does a SELECT on a relation containing only unclassified tuples. In the result, the tuples will all be labeled unclassified. However, if the user requests that the result be sorted, then a temporary relation must be created, and all tuples written to that relation will be labeled at the level of the subject. The result is that after the sort, all the tuples will be labeled secret. To avoid this problem, tuples written into temporary relations can be labeled at a user specified level that is strictly dominated by the level of the subject. This policy is safe because temporary relations are available only to the current process, and are destroyed upon close. The only subject that can possibly access a temporary relation is the subject that created it. This policy underscores the advisory nature of tuple level labels.
- *The replication of audit functionality.* In the monolithic TCB version of Trusted RUBIX, a single audit mechanism was used. In the subset version, both subsets need to do audit, but you need to ensure that the DAC TCB cannot interfere with the MAC TCB auditing. The solution that we used was that both subsets used similar mechanisms to write their audit records to the operating system audit trail. Since both can only add to the audit trail, the integrity of the audit trail is guaranteed. The mechanism for storing audit criteria for the two TCBs is independent.

7 Conclusions and Future Research

The document described the results of reengineering the Trusted RUBIX TCB architecture to incorporate the concepts of TCB subsetting as described in the Trusted Database Interpretation of the Trusted Computer System Evaluation Criteria (TCSEC). The following are the major lessons learned during this effort:

- A DBMS architecture that combines the concepts of trusted subjects and TCB subsets is technically feasible. This report described an architecture, with separate MAC and DAC subsets, where the MAC subset consists of the underlying operating system TCB, plus a layer of privileged DBMS code.
- The TCB subsetting approach can be used to significantly reduce the amount of code that requires MAC privilege in a Trusted DBMS. The architecture developed in this effort reduced the amount of code that runs with privilege by over 70 percent.
- The TCB subsetting approach used in this effort simplifies security analysis and therefore increases assurance. There are two primary reasons for this simplification. First, the interface to the MAC TCB is very simple and therefore the specification of the security semantics and the mapping to the formal model is simplified. This mapping is further simplified because the MAC policy itself is simple. Second, certain analyses need only be applied to subsets that implement a MAC policy (e.g., covert channel analysis). Since only the MAC subset implements a MAC policy, the amount of code that needs to be analyzed to establish these properties is substantially reduced.
- This effort substantiated the expectation that a subset architecture would incur in a performance and resource utilization penalty. Based on our benchmarks and the optimizations that are open to us, we believe that these degradations do not severely impact the viability of the architecture. It should also be noted that if the architecture was ported to an operating system that supported a more efficient domain isolation mechanism (e.g., protection rings [12]), the performance and resource utilization costs could be significantly reduced.

There are a number of directions in which this work can be extended. The first is performance engineering. As noted in Section 5.3.2, there are a number of potential design modifications that could significantly improve performance. A second area of future work is further minimization of the Trusted RUBIX kernel. As noted in Section 6, kernel minimization must be carefully weighed against other design goals (e.g., performance, flexibility). A final area of future research is in the area of DAC policy. The TCB subset architecture allows Trusted RUBIX to potentially support multiple DAC policies without impacting the MAC subset. Additional work could be done to specify and prototype different DAC mechanisms for Trusted RUBIX (e.g., assured DAC, role based DAC).

References

- [1] National Computer Security Center. Trusted database interpretation of the trusted computer system evaluation criteria. Technical Report NCSC-TG-021, National Computer Security Center, April 1991.
- [2] Department of Defense. Department of Defense trusted computer system evaluation criteria. DOD Standard 5200.28-STD, Department of Defense, December 1985.

- [3] Teresa F. Lunt and Peter K. Boucher. The SeaView prototype: project summary. In *Proceedings of the 17th National Computer Security Conference*, Baltimore, Maryland, October 1994.
- [4] J. P. O'Connor. Trusted RUBIX: A multilevel secure client-server DBMS. In *Proceedings of the Eighth Annual IFIP Working Group 11.3 Working Conference on Database Security*, August 1994.
- [5] G. J. Popek and C. S. Kline. A verifiable protection system. *Proceedings of the International Conference on Reliable Software*, pages 294-304, 1975.
- [6] M. Schaefer and R. R. Schell. Toward an understanding of extensible architectures for evaluated trusted computer system products. *Proceedings of the 1984 Symposium on Security and Privacy*, pages 41-49, 1984.
- [7] W. Shockley and R. R. Schell. TCB subsets for incremental evaluation. *Proceedings of the Third Aerospace Computer Security Conference*, December 1987.
- [8] J. P. Anderson. Computer security technology planning study. Technical Report ESD-TR-73-51 (AD-758206), J. P. Anderson Co., October 1972.
- [9] Unix System Laboratories. *System V Release 4.1 ES Network User's and Administrator's Guide*, 1991.
- [10] R. R. Schell, T. F. Tao, and M. Heckman. Designing the Gemsos security kernel for security and performance. *Proceedings of the 8th National Computer Security Conference*, 1985.
- [11] W. Timothy Polk and Lawrence E. Bassham III. Security issues in the database language SQL. NIST Special Publication 800-8, National Institute of Standards and Technology, August 1993.
- [12] M. D. Schroeder and J. H. Saltzer. A hardware architecture for implementing protection rings. *Communications of the ACM*, 15(3):157-170, March 1972.
- [13] T. Lunt, D. Denning, R. Schell, M. Heckman, and W. Shockley. Element-level classification with A1 assurance. *Computers and Society*, August 1988.
- [14] D. Bell and L. LaPadula. Secure computer system: Unified exposition and Multics interpretation. Technical Report MTR-2997, MITRE Corporation, July 1975.
- [15] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9), September 1975.
- [16] Infosystems Technology, Inc. TCB subset DBMS architecture project: Design document. Technical Report TR-9403-00-01, Infosystems Technology, Inc., December 1994.

- [17] Infosystems Technology, Inc. Trusted RUBIX formal model. Internal technical report, Infosystems Technology, Inc., July 1994.

***MISSION
OF
ROME LABORATORY***

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

- a. Conducts vigorous research, development and test programs in all applicable technologies;
- b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;
- c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;
- d. Promotes transfer of technology to the private sector;
- e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.